Summer School on Numerical Optimization Software - University of Freiburg

Exercise 6: Conic Programming

Gianluca Frison, Hans Mittelmann, Moritz Diehl

Semidefinite Programming Problem

An area that is relatively new is semidefinite programming (SDP). With it one can approximate many hard discrete optimization problems. The problems are continuous and have among the constraints some that prescribe the positive-semidefiniteness of a matrix variable.

In the following there are some references about SDP (that are not strictly necessary to solve the exercise):

`http://plato.asu.edu/abstracts/sdpasu.pdf`
`http://plato.asu.edu/kojima_sdp.pdf`
`http://www.optimization-online.org/DB_HTML/2001/07/358.html`
`http://www.optimization-online.org/DB_HTML/2010/08/2694.html`
`http://plato.asu.edu/ftp/sparse_sdp.html`

A convenient tool for solving SDPs and other convex problems phrased in Matlab is CVX, a Matlab-based modeling system for convex optimization

`http://cvxr.com/cvx/`

For this exercise, the free demo version (with non-commercial solvers SDPT3 and SeDuMi) will suffice.

Look at the slides of the talk

`http://plato.asu.edu/talks/mittelmann_sdp.pdf`

On slide #11 it is shown that every real matrix B can be split as the sum of a positive semidefinite and a negative semidefinite matrix, as

$$B = B^+ - B^-, \quad B^+, B^- \succeq 0.$$

Let $Y^+ = XB^+X^T$ and $Y^- = XB^-X^T$, we have

$$Y^+, Y^- \succeq 0.$$

On slide #12 of the talk a basic SDP relaxation (relaxation because the integer constraint on the matrix X is relaxed) is given:

$$\begin{aligned}
\min \quad & \mathrm{Tr}(A(Y^+ - Y^-)) \\
s.t. \quad & Y^+ e = XB^+ e \\
& Y^- e = XB^- e \\
& \mathrm{diag}(Y^+) = X\mathrm{diag}(B^+) \\
& \mathrm{diag}(Y^-) = X\mathrm{diag}(B^-) \\
& Y^+ \geq \min(B^+) \\
& Y^- \geq \min(B^-) \\
& Y^+ - XB^+X^T \succeq 0 \\
& Y^- - XB^-X^T \succeq 0 \\
& Xe = X^T e = e \\
& X \geq 0
\end{aligned}$$

where Tr is the trace (i.e. the sum of the diagonal elements of a square matrix).

6.1 Phrase this minimization problem in Matlab for use in CVX and compute lower bounds for three quadratic assignment problems (QAPs) in

`http://anjos.mgi.polymtl.ca/qaplib/inst.html`

namely, NUG12 (data in `nug12.dat`), NUG30 (data in `nug30.dat`) and TAI30A (the smallest unsolved problem in QAPLIB, data in `tai30a.dat`). Compare to the bounds given in QAPLIB at the above link.

Note that you can exchange the A and B matrices and use the better bound. How much worse do the bounds get demanding that just the Y^+ and Y^- matrices are PSD, in place of $Y^+ - XB^+X^T$ and $Y^- - XB^-X^T$?

Hint: some CVX notation. We will use the SeMiDu solver in CVX, that can be chosen with the option

```
cvx_solver sedumi;
```

All CVX models must be preceded by the command `cvx_begin` and terminated with the command `cvx_end`. All variable declarations, objective functions and constraints should fall in between. Variables are declared using the `variable` command. E.g. the variable matrix X of size $n \times n$ is declared as

```
variable X(n,n)
```

Variable can also include one or more keywords to denote various structures or conditions on the variable. E.g. the variable symmetric matrix Y^+ of size $n \times n$ is declares as

```
variable Yp(n,n) symmetric
```

The declaration of an objective function requires the use of the `minimize` or `maximize` functions. E.g. the objective of the problem to be solved is

```
minimize( trace( A*(Yp-Ym) ) )
```

Constraints are declared after a `subject to` keyword, and can be either equality ($==$), less-than ($<=$) or greater-than ($>=$) constraints. E.g. the conditions that all the elements of the matrix X must be non-negative and that the matrix Yp must be positive semidefinite are

```
subject to
    X >= 0
    Yp == semidefinite(n)
```

Hint: In order to split a matrix B as $B = B^+ - B^-$ with B^+ and B^- positive semidefinite, you can use the `eig` command in Matlab to compute e.g. B^+ as

```
[V,D] = eig(B);
Dp = max(D, zeros(n));
Bp = V*Dp*V';
```

and similarly for B^- (beware the sign of eigenvalues!).

Hint: standard SDPs are linear and our relaxation has quadratic SDP constraints. You can linearize them by taking the matrix square root R^+ of B^+ (and similarly the matrix square root R^- of B^-) and by replacing the inequalities

$$Y^+ - XB^+X^T \succeq 0, \quad Y^- - XB^-X^T \succeq 0$$

by

$$\begin{bmatrix} I & (Z^+)^T \\ Z^+ & Y^+ \end{bmatrix} \succeq 0, \quad \begin{bmatrix} I & (Z^-)^T \\ Z^- & Y^- \end{bmatrix} \succeq 0$$

where $Z^+ = XR^+$ (and similarly $Z^- = XR^-$). Note that you can exploit the construction of B^+ and B^- in the previous hint to build R^+ and R^- from the matrix square root of `Dp` and `Dm`. Also note that the matrices `Dp` and `Dm` are diagonal, so the matrix square root is equivalent to the element-wise square root (if Matlab or Octave complain about something).

Second Order Cone Programming Problem

This exercise is based on the LP formulation and the two robust formulations in `robust.pdf`. One of the two robust formulations leads to a second order cone programming (SOCP).

Consider the following LP

$$\begin{aligned} \max_x \quad & c^T x \\ s.t. \quad & Ax \le b \\ & l \le x \le u. \end{aligned}$$

In the above formulation it is assumed without loss of generality that data uncertainty only affects the elements in matrix A. Each entry a_{ij} is modeled as a symmetric and bounded random variable $\tilde{a}_{ij}$ that takes values in $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$.

The robust formulation of Soyster is (2.1)

$$\begin{aligned} \max_x \quad & c^T x \\ s.t. \quad & \sum_j a_{ij} x_j + \sum_{j \in J_i} \hat{a}_{ij} y_j \le b_i \qquad \forall i \\ & -y_j \le x_j \le y_j \qquad \forall j \\ & l \le x \le u \\ & y \ge 0. \end{aligned}$$

Let x^* be the optimal solution. At optimality it holds $y_j = |x_j^*|$, and therefore

$$\sum_j a_{ij} x_j^* + \sum_{j \in J_i} \hat{a}_{ij} |x_j^*| \le b_i \quad \forall i.$$

The idea is that for every i-th constraint, the term $\sum_{j \in J_i} \hat{a}_{ij} |x_j^*|$ gives the necessary protection by maintaining a gap between $\sum_j a_{ij} x_j^*$ and b_i.

Soyster's method provides the highest protection, but it is also the most conservative in the sense that the objective function value is much worse that the objective in the original LP. Ben-Tal and Nemirovski propose the robust problem (2.2)

$$\begin{aligned} \max_x \quad & c^T x \\ s.t. \quad & \sum_j a_{ij} x_j + \sum_{j \in J_i} \hat{a}_{ij} y_j + \Omega_i \sqrt{\sum_{j \in J_i} \hat{a}_{ij}^2 z_{ij}^2} \le b_i \qquad \forall i \\ & -y_j \le x_j - z_{ij} \le y_j \qquad \forall i, j \in J_i \\ & l \le x \le u \\ & y \ge 0. \end{aligned}$$

The authors have shown that the probability that the i-th constraint is violated is at most $\exp(-\Omega_i^2/2)$. The robust model (2.2) is less conservative that the model (2.1).

6.2 Consider the following problem.

A farmer has some cows and sheep. Stables are available for 50 cows and 300 sheep. The pasture is 72 acres. A cow needs 1 acre and a sheep 0.2. Labor is available for 10,000 hours per year with a cow requiring 150 hours and a sheep 25. The farmer's profit is \$250 per cow and \$45 per sheep. Maximize the total profit.

Phrase in AMPL the LP and the robust LPs as in the writeup (2.1) and (2.2). Assume the number of cows and sheep to be a real variable (not integer). Assume a perturbation of 5% in the A matrix. By how much is the objective perturbed for the two robust models? Use $\Omega = 0.1$ for (2.2).

Why is (2.2) called a SOCP problem?